INTL-0545-US
(P11071)

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: **FIRMWARE UPGRADE USING ADDRESS CONVERSION**

INVENTORS: **JOHN P. LAMBINO, JOHN V. LOVELACE, DAVID I. POISNER, AND ANDREW W. MARTWICK**

Express Mail No.: EL669040513US

Date: May 22, 2001

FIRMWARE UPGRADE USING ADDRESS CONVERSION

Background

This invention relates to firmware for a processor-based system, and, more particularly, to a fail-safe mechanism for performing firmware upgrades.

A processor-based system typically includes firmware for initializing the system. Firmware is a software program that is permanently or semi-permanently resident in the processor-based system. Usually, the software program is "burned" into a read-only memory (ROM) or a flash memory device. The ROM or flash devices may be removable integrated circuits (ICs) that plug into a dedicated chip slot in the system board.

Although the device storing the firmware may be removable and, thus, physically replaced, more typically, the device is re-programmed in place, e.g., without physical removal. ROMs may be programmable (PROMs), erasable (EPROMs), and electrically erasable (EEPROMs), such as flash memory. Flash memory is also programmable, and may typically be programmed at a faster rate than other EEPROMs.

Like other software, the firmware itself is a valuable component of the processor-based system. Firmware is the very first code executed in the system. The firmware initializes the key hardware components. Once the system is initialized, the firmware typically loads an operating system loader program into memory. The loader program then loads the operating system.

The firmware comprises part of the identity of the processor-based system. Many computer manufacturers, for example, include a proprietary firmware that includes features and capabilities that may distinguish the processor-based system from those of other manufacturers.

In many systems, the firmware is divided up where only portions of the firmware may be upgraded. This assures that, despite interruption on the firmware upgrade, a minimum amount of firmware is available to power on the system.

Sometimes, however, the non-upgradable portion of the firmware needs upgrade as well. In such cases, more draconian measures, such as physical removal of the firmware device, may be the only way to perform such upgrades.

Thus, there is a continuing need to provide a fail-safe mechanism for upgrading the firmware.

Brief Description of the Drawings

Figure 1 is a block diagram of a system according to one embodiment of the invention;

Figure 2 is a block diagram of a firmware configuration according to one embodiment of the invention;

Figure 3 is a diagram illustrating upgrade of the boot block according to one embodiment of the invention;

Figure 4 is a flow diagram of the firmware upgrade of Figure 3 according to one embodiment of the invention;

Figure 5 is a diagram illustrating upgrade of the boot block according to a second embodiment of the invention;

Figure 6 is a flow diagram of the firmware upgrade of Figure 5, according to the second embodiment of the invention; and

Figure 7 is a component layout of the system according to one embodiment of the invention.

Detailed Description

According to one embodiment, a system may successfully upgrade a boot block portion of a firmware program, as described below. In Figure 1, a system 100 includes a processor 10 and a flash memory 20 for storing a boot block program 26. The boot block 26 is part of a firmware program, executed by the processor 10 when the system 100 powers on.

In one embodiment, the boot block 26 performs minimal initialization of the system. Typically, the boot block 26 is not upgradable. A typical firmware configuration 70 is illustrated in Figure 2. The boot block 60, which is typically not upgradable, is followed by several upgradable blocks 62. The boot block 60 and the upgradable blocks 62 may themselves include one or more blocks (not shown).

When executed, the boot block 60 initializes the system 100 such that the other blocks 62 of the firmware program 70 may be upgraded. As will be shown below, in addition to the blocks 62 being upgradable, the boot block 60 itself may also be securely updated.

In one embodiment, the flash memory 20 includes a primary location 22 and a secondary location 24. As shown, the boot block 26 initially resides in the primary location 22. The primary location 22 and the secondary location 24 facilitate upgrade of the boot block 26, without requiring the system 100 to maintain power during the upgrade. In other words, the system 100 has a fail-safe mechanism for successfully upgrading the boot block 26, even when a power loss occurs.

In one embodiment, the system 100 further includes address conversion 30. As shown in Figure 1, address conversion 30 enables the processor 10 to alternate between executing software located in either the primary location 22 or the secondary location 24. In one embodiment, address conversion 30 consists

simply of toggling a single bit, such that a different address is accessed by the processor 10 during execution.

The address conversion 30 is coupled to a backup battery 34. In one embodiment, the backup battery 34 maintains the state of a bit that was toggled by the address conversion 30, following power off of the system 100. The backup battery 34 may be similar to those used for maintaining semi-volatile memory such as complementary metal oxide semiconductor (CMOS) memory.

In one embodiment, the system 100 further includes a jumper 32, coupled to the address conversion 30. The jumper 32 provides another mechanism for toggling the address bit, used to switch between the primary location 22 and the secondary location 24 of the flash memory 20.

The system 100 also includes a non-volatile storage 38, such as a hard disk drive. In one embodiment, the non-volatile storage 38 holds a new boot block 36. The new boot block 36 is a replacement boot block for the boot block 60 in the flash memory 20.

The new boot block 36 may be received to the system 100 from a network 40. Accordingly, the system 100 includes a network interface card (NIC) 42. The new boot block 36 may also be received from a floppy drive (not shown).

The system 100 also includes a memory 44, for storing operating system and other software, including a software program 200. In one embodiment, the software program 200 performs the steps, described below, for upgrading the boot block 60 with the new boot block 36. In other embodiments, the boot block 60 itself performs the upgrade. For simplicity, the software 200 is depicted as a separate program from the boot block 60.

In Figure 3, a diagram illustrates six steps or operations performed by the system 100 to upgrade the boot block 60. Each succeeding operation is specified by a numeral, 1 to 6. The primary location 22 and secondary location

24 of the flash memory 20 are shown. As in Figure 1, the boot block 60 occupies the primary location 22. In one embodiment, the primary location is addressed at 0ffffff0h. When the system 100 receives power, execution of the firmware 70 proceeds from this address.

In one embodiment, the primary location 22 of the flash memory 20 occupies 64K of the flash memory. Accordingly, the secondary location 24, which is adjacent to the primary location, is addressed starting at 0fffeff0h, as shown in Figure 3. The size of the primary and secondary locations as well as the address from which they are accessed are illustrative and should in no way be construed as limiting.

Initially, according to one embodiment, the processor 10 executes from the higher address, e.g. 0ffffff0h (Step 1). At this point, the new boot block 36 is retrieved from the non-volatile storage 38 and stored in the secondary location 24 (Step 2). The execution address remains pointed at the primary location 22.

In one embodiment, the new boot block 36, now in the secondary location 24, may be executed simply by changing the execution address. The address conversion 30 toggles a bit such that the processor 10 executes from the secondary location address, e.g. 0fffeff0h (Step 3). Notice that the location of the boot block 60 has not changed at all.

With the execution address still pointing at the secondary location 24, the new boot block 36 is copied from the secondary location 24 to the primary location 22 (Step 4). Because the processor 10 is now executing from the secondary location 24, the old boot block 60 is no longer needed. The new boot block 36 thus replaces the old boot block 60.

In one embodiment, before proceeding further, a confirmation is made that the new boot block 36 was successfully copied to the primary location 22 (Step 5). Notice that the new boot block 36 is now located in both the primary

location 22 and the secondary location 24. Notice also that the execution address remains pointed to the secondary location 24.

Once assurance that the new boot block 36 was successfully copied to the primary location 22, the execution address is changed back to point to the primary location 22 (Step 6). The address conversion 30 changes the address from which the processor 10 executes instructions. In the embodiment of Figure 3, the execution address may be changed by simply toggling a single address bit, A16. Note that the new boot block 36 is now located in the primary location 22. The secondary location 24 also still has the new boot block 36. However, execution by the processor 10 does not take place from the secondary location 24. Thus, the contents of the secondary location 24 are irrelevant.

The software 200 may perform the operations of Figure 3, as depicted in the flow diagram of Figure 4. According to one embodiment, the new boot block 36 is first copied into the secondary location 24 (block 202). The execution address is then pointed to the secondary location 24, such as by toggling an address bit (block 204). This causes the new boot block 36 to be executed rather than the old boot block 60. If power is removed from the system 100, the system 100 continues to execute from the secondary location 24 until either the backup battery 34 or the jumper 32 are adjusted, in which case execution shall proceed from the primary location 22.

Next, the new boot block 36 is copied from the secondary location 24 to the primary location 22 (block 206). Before toggling the execution address back to the primary location 22, however, the software 200 confirms that the new boot block 36 got the primary location 22 in its entirety (block 208). Should the transfer of the new boot block 36 have been interrupted for some reason, the execution address continues to point to the new boot block 36 in the secondary location 24, so the system will still boot properly. Once the confirmation is complete, the execution address is modified to point back to the primary location

22 (block 210). In one embodiment, changing the execution address is achieved by toggling an address bit.

In Figure 5, a block diagram illustrates operation of the system 100 to upgrade the boot block 60 according to a second embodiment. As in Figure 3, each succeeding step or operation is specified by a numeral, 1 to 6. The primary location 22 and the secondary location 24 of the flash memory 20 are shown. As in Figure 1, the boot block 60 occupies the primary location 22 and is addressed at 0ffffffff0h. When the system 100 receives power, execution of the firmware 70 proceeds from this address. The secondary location 24, likewise, is addressed starting at 0fffefff0h.

First, according to one embodiment, the processor 10 executes from the 0ffffffff0h address (Step 1). The contents of the secondary location 24 are at this time unknown. Then, rather than copying a new boot block from the memory 44, the boot block 26 is copied from the primary location 22 to the secondary location 24 (Step 2). The execution address remains pointed at the primary location 22.

Once the boot block 26 is stored in the secondary location 24, the execution address may be changed to point to the secondary location 24 (Step 3). The address conversion 30 toggles the bit such that the processor 10 executes from the secondary location execution address e.g., 0fffefff0h.

With the execution address still pointing at the secondary location, the new boot block 36 is copied from the non-volatile storage 38, first to the memory 44, then to the primary location 22. Alternatively, the new boot block 36 may be downloaded from across the network 40 to the system 100, in its memory 44. The new block 36 may then be copied to the primary location 22 (Step 4).

Once the new boot block 36 is completely copied to the primary location 22, the execution address may be changed back to point to the primary location 22. In one embodiment, however, a confirmation is made that the new boot

block 36 was successfully copied to the primary location 22 (Step 5). In step 5, the execution address remains pointed to the secondary location 24.

Once assurance that the new boot block 36 was successfully copied to the primary location 22, the execution address is changed back to point to the primary location 22 (Step 6). The address conversion 30 changes the address to which the processor 10 will execute instructions. In the embodiment of Figure 5, as in Figure 3, the execution address may be changed by simply toggling a single address bit, A16. Note that the new boot block 36 is now located in the primary location 22. The contents of the secondary location 24 are irrelevant.

The software 200 may perform the operations of Figure 5, as depicted in the flow diagram of Figure 6. According to one embodiment, the boot block 26 is copied from the primary location 22 into the secondary location 24 (block 222). The execution address is then modified such that it points to the secondary location 24 (block 224). In one embodiment, the address conversion 30 of the system 100 changes the execution address. Address conversion causes the boot block 26 to be executed from the secondary location 24 rather than from the primary location 22. If power is removed from the system 100, the system 100 may continue to execute from the secondary location 24 until either the backup battery 34 or the jumper 32 is adjusted.

Next, the new boot block 36 is copied to the primary location (block 226). The new boot block 36 may be loaded into the memory 44 prior to this step, then copied to the primary location 22. In one embodiment, the software 200 confirms that the new boot block 36 got to the primary location 22 in its entirety prior to changing the execution address (block 228). Where transfer of the new boot block 36 failed or was otherwise interrupted, the execution address remains pointing to the secondary location 24. This ensures that the system still boots properly. Once the confirmation is complete, the execution address is modified to point back to the primary location 22 (block 230).

In one embodiment, the system 100 is part of a processor-based system, as depicted in Figure 7. The processor 10 is coupled to a host bus 48, which connects the processor to the rest of the system 100. A bridge 46 is coupled between the host bus 48 and a PCI bus 52, according to one embodiment. The memory 44 is connected to the bridge 46.

In one embodiment, the system also includes a south bridge 50. The south bridge is a multi-function bridge, which supports the flash 20, the network interface card 42, and the non-volatile storage 38, as well as the battery 34 and jumper 32.

The above operations may be implemented where distinct devices share a single address. In some systems, multiple devices may each be addressed by the same address. The system typically includes a device select or other logic mechanism for deciding which of the multiple devices is to be accessed when the processor executes from the shared address.

In such systems, rather than move the execution address as done by the address conversion 30, described above, the device select or other logic mechanism may be adjusted. While no address conversion takes place in such systems, the principal of moving where the processor executes remains. Other mechanisms for alternately selecting devices may likewise be exploited in implementing a firmware, device driver, or other software upgrade.

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.